REMARKS/ARGUMENTS

Spelling errors and typographical errors have been corrected in claims 2, 6, 8, 13, 19, 24, 28, 30, 35, 36, 39, 40, 43, and 44. These corrections were not caused by prior art.

In paragraph 2, the examiner has objected to claim 7 for the use of the word "proceeding" and has suggested the word "preceding". The examiner is assuming this interpretation in the action. The examiner's objection is traversed. Claim 7 claims the internal method of a Transaction Method when executing Business Application messages. An execution unit starts with an interactive transaction record and either ends with the next interactive transaction record, or with a last preemptive transaction record in the User Interaction Graph. The term "preceeding" is use in the context of the Transaction Method and expresses the process of executing the Transaction Method. The term "succeeding" used later in the same sentence is used in the context of a Transaction Record to express that one or more preemptive transaction records which succeed (follow next in order) the interactive transaction record while processing the Transaction Method. Claims 7, 18, and 29 have been amended to make clearer that the Transaction Method proceeds with the execution of Business Application messages. These amendments were not caused by prior art. Allowance of claims 7, 18 and 29, as amended, is requested.

In paragraphs 3 and 4, the examiner rejects claims 1, 4, 5, 9, 12, 15, 16, 20, 23, 26, 27, 31 and 34-45 under 35 U.S.C. 102(b) as being anticipated by White, U.S. Patent 5,428,782. The rejections are traversed, and reconsideration is requested.

White is directed to enabling a platform independent development of traditional transaction applications (SQL based) by providing tools and formats to define Data formats of procedures (views), panel designs (maps) and coding of business logic procedure body. From these elements a code generator generates the code. The Transaction Definition Table assures that during runtime, the correct elements are accessed to run the transactions. White in fact, solves the inverse problem of the present invention. In White, a programmer's work is limited to manually program processing logic (see col. 3, lines 67 and col. 8, lines 37 to 45). White provides a single source code version of an application which can be used to build by compilation, different executables (load modules) for execution on different target computer architectures, that is, different execution environments (col. 6, lines 13-21).

These functions in White are solved by:
1. separating the user interface logic (primarily based on IET) from the application logic (Col. 8, lines 37 - 45; col. 7, lines 22-53; Figs. 5 and 5, and Col. 19, lines 20-25), and
2. allowing the user to define the various panels and corresponding panel flow including all input and output data elements and of compiling these dialogue related definitions into an executable form, which can be understood by the IET performing the panel navigation (col. 8, lines 5 - 24; and col. 8, lines 47-67).

White discloses that although a terminal has a human interface for initiating transactions, a "terminal" could also be a program, which could process a sequence of panels autonomous of human interaction (col. 19, lines 42-54). White discloses applications on a level of a transaction application test tool

which is running manually reconfigured interaction scenarios automatically.

In the present invention, one side has business objects defined for the client application which contain attributes needing data from the legacy database (legacy attributes). On the other side, there is an existing transaction system which contains all business logic to read and write the needed data correctly. Normally, legacy attributes from one panel will be scattered in several business objects because design principles of 00 applications and legacy transaction systems are very different. In the present invention data items from the legacy panels are assigned to business object attributes, and transaction sequences for getter/setter methods are automatically computed for these attributes. The sequences for the legacy application is generated by the invention and not configured manually.

Claims 1, 12, 23, 34 and 42 have been amended to make clear that 'sequence' is claimed as a sequence of Business Application panels within one execution of a Business Application.

With regard to claim 1:

In the analysis step, in White, the TDT is used during runtime of a to-be-constructed application to interact with the IET. The analysis step performed by our tool refers to existing Business Applications. Our analysis step takes place after the construction of a Business Application to analyze the panels of said Business Application to determine their input data in order to generate access to them (see the preamble).

In the generating step, White discloses a GTD generating compiled view objects from the TDF, not the TDT (col. 8, lines 37-41). The IET uses the TDT at runtime, not during the generation step. Furthermore, the disclosed GDT of White does

not produce application specific logic. Col. 8, lines 51-52 states that the usage of the IET (based on TDT) enables the application procedure to contain primarily application specific logic. The application specific logic as disclosed by White has to be written by a programmer, and is not generated by the GDT (col. 9, lines 27-28).

In "wherein said generation step generates program code into the procedure, ... is providing required input data according to said analysis step to at least one sequence of succeeding Business Application panels, said sequence comprising at least one Business application panel," the analysis step performed by the invention refers to an existing Business Application (a transaction), whereas the TDT from White is generated by the GDT out of the TDT for a new application (a set of transactions). In White, the TDT is the outcome of the GDT and cannot be compared to an analysis step. The program code generated by the present invention of claim 1 is able to execute at least a sequence of Business Application panels. Claim 1 has been amended to make clear that 'sequence' is claimed as a sequence of input/output Business Application panels within one execution of a Business Application. In White, col. 9, lines 28-34, there is multiple executions of the same transaction.

In "wherein said generating step generates ... is performing a required activity for launching said Business Application to autonomously process at least one sequence of succeeding Business application panels without interaction with said user," White discloses the fact that a terminal need not be a human interface, but can be an interactive program. However, White says nothing about the structure of the program processing the sequence of panels autonomously. The generation step performed by the present invention does generate code to allow a Business Application to be launched without human interaction.

In regard to claim 5, White teaches that a programmer uses the GTD menus to update the TDF to construct a transaction, that is compile and link the correct components to generate the transaction itself. Claim 5 teaches the generation of Business Application message descriptions form existing Business Application message implementations.

In regard to claim 9, the transaction method controls the execution of at least a sequence of an existing Business Application.

In regard to claims 12, 16, 20 (computer apparatus) and claims 23, 27, 31 (program storage device), see the remarks regarding the corresponding claims 1, 5, 9 (method).

In regard to Claim 34;

In "a transaction method called from a program ... without interacting with said user," the TDT from White is generated by the GDT out of the TDT for a new application (a set of transactions). The disclosed GDT of White does not produce application specific logic. Col. 8, lines 51-52 states that the usage of the IET (based on TDT) enables the application procedure to contain primarily application specific logic. The application specific logic as disclosed by White has to be written by a programmer, and is not generated by the GDT (col. 9, lines 27-28).

In "wherein said transaction method is autonomously providing required input data ... to at least one sequence of succeeding Business Application panels," the program code called by the present invention of claim 34 is able to execute at least a sequence of Business Application panels. The term 'sequence' used in claim 34 refers to a sequence of input Business Application panels within one execution of a Business

Application. In White, col. 9, lines 28-34, there is multiple executions of the same transaction.

In "wherein said transaction method if performing the required activity for launching said Business Application or process ... a next Buenos Application panel .. without interaction with said user," White discloses the fact that a terminal need not be a human interface, but can be an interactive program. However, White says nothing about the structure of the program processing the sequence of panels autonomously. The Transaction Method of claim 34 is performing the required activity to allow a Business Application to be launched without human interaction.

In regard to claim 35,

In "wherein said transaction method includes handling Business Application messages with respect to individual Business Application panel elements based upon transaction records having Business Application message descriptions," White teaches in col. 8, lines 37-50 how to generate views to be used to process I/O data of a transaction panel independent form the platform the user is working with. Claim 35 depended from claim 34 claims that the transaction method executes several succeeding panels and uses data from the panels.

In "wherein said Transaction Method launches the Business Application to process ... a next Business Application panel according to a Business Application message transition action." White teaches the linking from one application to another application and to maintain the state within a view stack. In claim 35, the processing of a next Business Application panel is within the same Business Application. How to perform the transition in included (coded) in the Transaction Method, and not part of a view stack.

In regard to claims 38-39 (a computer apparatus) and claims 42-43 (program storage device), see the remarks regarding the corresponding claims 34 and 35 (method).

It is respectfully submitted that claims 1, 4, 5, 9, 12, 15, 16, 20, 23, 26, 31, 34-45 are allowable under 35 U.S.C. 102(b) which allowance is respectfully requested.

In paragraphs 5 and 6, claims 2, 3, 6-8, 13, 14, 17-19 24, 25, and 28-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over White, U.S. Patent 5,428,782 as applied to claims 1, 12, and 23 respectively, and further in view of Chen et al., U.S. Patent 5,806,062. The rejection is respectfully traversed and reconsideration is requested.

Chen et al. describes a system comparing an analyzing documents. These documents can be html texts, source code, or any other coded information. Chen discloses a method to describe the differences of two documents in a directed graph. Chen discloses only the automatic creation and display of this graph. In his invention, the graph is not processed automatically, but is used to visualize the differences described. The manual query is supported by the graph interface.

In the present invention, a graph describing the transaction flows in entered by the user in a graphical interface. Each node is a panel. The detailed information about the data in this panel is obtained by parsing the IMS or CICS description of the panels. Our designed framework automatically searches paths in the graph for receiving or writing all data elements connected to a business object. If a path is found, this information is used to generate the end point of a first TOM and the starting point of the next TOM automatically (see page 32).

In regard to claim 2:

In "further comprising generating said description of said Business Application panels prior to said analysis step ... being independent from the system environment," claim 2 comprises the method of generating system independent description for panels of already existing Business Applications. Transaction panels for heterogeneous systems are not built, but a description for Business Applications panels are generated that are independent from the underlying system environment, e.g. and IMS panel is abstracted for further usage within a Transaction Method. The GTD/TDF generation step is used to initially construct an application. The TDF is the blueprint for construction a DAA transaction.

In "a Business Application panel modeling step, in which at least one of said Business Application panels ... is modeled with respect to individual Business Application panel elements," during the modeling step the data elements of an existing transaction panel is parsed and prepared for further execution during the analyzing and code generation steps. White generates view objects to be used to process I/O data of a transaction panel independent for the platform the user is working with.

In "a Business Application message description generation step, in which at least one of said Business Applications Panels ... is parsed with respect to individual Business Application elements," col. 19, lines 1-6 of White cited by the examiner says "identifying the function key number, and "prompt" is variable length indication of the associated function. When using DL/1 or other data bases, it is desirable, or often even necessary, to have a collection of functions and sub-functions packaged as TRANA and while another set of functions are packaged as TRANB to". This language does not teach or suggest that at least one Business Applications Panels is parsed with respect to individual

Business Application elements. In the present invention, the parsing step is used to speed up the modeling step.

In "in which for each modeled Business Application message transaction record is generated storing said Business Application message Description," White describes the saving of status data from a procedure during runtime of the transaction application into a ROLLFILE. The modeled Transaction record does not store panel information during runtime but contains the information about the panel data formats.

In "a User Interaction Graph generation step, in which a sequence of Business Application messages ... is stored in a least one directed User Interaction Graph," Col. 17, lines 59-65 of Chen cited by the examiner teaches that the usage of directed graphs for the purpose of visualization of difference in data from documents. User interaction is allowed during the visualization and additional queries can be initiated. The claimed User Interaction Graph is used to formalize possible user interaction flows (not data) between a user and an existing Business Application. Thus, even if White and Chen were combined, as suggested by the examiner, the invention of claim 3 would not result.

In regard to Claim 3;

In "a Business Application message transition generating step ... after a current Business Application panel according to the sequence within the User Interaction Graph," col. 16, lines 42-47 and col. 137, line 51 to col. 138, line 3 of White cited by the examiner teaches how to obtain data for Application A form Application B on a different system in a transparent way for the user of Application A. This can only be done by using a view-stack. However, this is not required in the present invention for a Business Application Message transition, as such a message transition always stays within the same Business

Application, thus no view stack is needed. As opposed to White, the claimed generated message transition contains all the required code and data.

In "in which said Business Application message transition action is stored within the generated transaction record ..." the view stack taught by White can only be used to keep the current view (status) of the executed application transaction. Transitions between views are not kept in the view stack, they are part of the procedure logic. The claimed transaction record has to keep the required action to proceed to the next Business Application message, thus the view stack teaches away from the invention of claim 3.

In regard to claim 6:

In "message description generation step incorporates indications on specific characteristics into the generated transaction record encompassing

"an entry transaction record indication ... required for an initial start of a Business Application execution," col. 58, lines 25-66 of White cited by the examiner teaches invoking a DAA procedure being part of an application, not initiating a transaction. The characteristics taught by White cannot be used to initiate an existing transaction, and thus teach away from the invention of claim 6. An entry transaction record indication identifies the transaction record as the one to initiate the Business Application (transaction).

In "an external trisection record indication, characterizing a Business Application message being part of a second user interaction graph ...," in the invention of claim 6, an external transaction record is used to initiate another transaction, not to call to another procedure within the same or another application as in White. In the present invention, when initiating another transaction, no view stack is updated. In the

invention of claim 6, the processing is synchronous. What is expected back is part of the transaction method of the external transaction record. The view stack of White, as well as the calling of other procedures is not related to the claimed external transaction records.

In "an interactive transaction record indication, indicating a Business Application message on an execution unit of succeeding ...," Col. 17, lines 59-65 of Chen cited by the examiner teaches that the usage of directed graphs for the purpose of visualization of difference in data from documents. User interaction is allowed during the visualization and additional queries can be initiated. The claimed User Interaction Graph is used to formalize possible user interaction flows (not data) between a user and an existing Business Application. Thus, even if White and Chen were combined, as suggested by the examiner, the invention of claim 6 would not result.

In "a preemptive transaction record indication ... and not being a first Business Application message in said execution unit," a preemptive transaction record is a record that does not require any input parameters or interaction. Processing within the Transaction Method simply continues with the next Transaction Record. The view stack of White must be updated or restored, and thus teaches away from the invention of claim 6.

In regard of claim 7;

In "said transaction method is starting ... with an interactive transaction record," White refers to a procedure called by a panel. It receives input data from this panel. Thus, the Transaction Method is never called from a Transaction Record as claimed, but controls the flow through one or more panels. Thus in White, the starting point of a transaction method is always an interactive panel.

In "said truncation method is proceeding ... with all preemptive transaction records succeeding said interactive transaction record within the user interaction graph," a preemptive transaction record is a record that does not require any input parameters or interaction. Processing within the Transaction Method simply continues with the next Transaction Record. Therefore, no view stack has to updated or restored, as in White. White thus teaches away from the invention of claim 7.

In "said transaction method is ending ... with a next interactive transaction record or with a last preemptive transaction record if no transaction record is succeeding...," a preemptive transaction record is a record that does not require any input parameters or interaction. Processing the Transaction Method simply continues with the next Transaction Record. As previously pointed out, in the present invention, no view stack needs to be updated or restored as required in White.

In regard to claim 8;

In "encompasses as transaction method input parameters ...," the GTD of White is used to construct an application whereas the claimed Transaction Method is used to access existing Business Applications. The GTD of White defines single transitions (input-procedure-output), whereas the claimed Transaction Method encompasses a multitude of input and output parameters for all preemptive Transaction Records handled by the Transaction Method.

In "encompasses as transaction method output parameters ...," the GTD of White is used to construct an application whereas the claimed Transaction Method is used to access existing Business Applications. The GTD of White defines single transitions (input-procedure-output), whereas the claimed Transaction Method encompasses a multitude of input and output parameters for all preemptive Transaction Records handled by the Transaction Method.

In regard to claims 13-14 and 17-19 (computer apparatus) and claims 24-25 and 28-30 (program storage device), see the remarks regarding the corresponding claims 2-3 and 6-8 (method).

It is respectfully submitted that claims 2, 3, 6-8, 13, 14 17-19, 24, 25, and 28-30 are allowable over White in view of Chen for the reasons set out above, and allowance is respectfully requested.

Claims 10, 11, 21, 22, 32, 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over white as applied to claims 1, 12, and 23, respectively, and further in view of Khalidi. The rejection is respectfully traversed, and reconsideration is requested for the same reasons as given in the remarks of claims 1, 12, and 23 above.

Further, Khalidi shows how to implement new transaction applications in a object oriented manner.

In the present invention, legacy transaction applications are implemented on a host. These legacy applications contain a lot of knowledge about business logic. In the present invention, a framework is claimed to make these existing legacy transaction available from new object oriented client applications with as few coding work as possible.

Is respectfully submitted that claims 10, 11, 21, 22, 32, and 33 are allowable under 35 U.S.C. 103(a) over White in view of Khalidi for the reasons set out above, which allowance is respectfully requested.

It is respectfully submitted that the application is now in condition for allowance, which allowance is respectfully requested.

RESPECTFULLY SUBMITTED

BY: FLOYD A. GONZALEZ-Attorney
Registration No. 26,732
Phone: 845-433-1163
Fax: 845-432-9786